In the Claims:

Please amend claims 1, 5-6, 8-9, 19, 21, 23-24, 26, 36, 38-39, 41, 53-54, 56, and 66-67, as indicated below.

1. (Currently amended) A method implemented in a device supporting a public-key cryptography application, the method comprising:

a first arithmetic circuit comprising a first plurality of arithmetic structures feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set in the public-key cryptography application, generated by the first arithmetic circuit, to a second arithmetic circuit comprising a second plurality of arithmetic structures;

the second arithmetic circuit generating a first partial result of a currently executing single arithmetic instruction of the processor instruction set in the public-key cryptography application, wherein the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result representing the high order bits summed with low order bits of a result of a first number multiplied by a second number, the summing of the high order bits being performed during multiplication of the first number and the second number, the summing and at least a portion of the multiplication being performed in the second arithmetic circuit;

storing the first partial result; and

using the stored first partial result in a subsequent computation in the public-key cryptography application.

2. (Original) The method as recited in claim 1 wherein the high order bits are fed back in redundant number representation.

3. (Original) The method as recited in claim 2 wherein the redundant number representation includes sum and carry bits.

4. (Previously presented) The method as recited in claim 1, further comprising feeding back the high order bits through a register to the second arithmetic circuit.

5. (Currently amended) The method as recited in claim 1, further comprising:
generating a second partial result of the currently executing single arithmetic instruction in the first arithmetic circuit, the second partial result representing the high order bits of the multiplication result of the first number multiplied by the second number.

6. (Currently amended) The method as recited in claim 1, further comprising:
generating a second partial result of the currently executing single arithmetic instruction, the second partial result representing the high order bits of the multiplication result of the first number multiplied by the second number summed with the high order bits of the previously executed single arithmetic instruction.

7. (Original) The method as recited in claim 6 further comprising supplying values generated in one or more most significant columns of the second arithmetic structures to one or more least significant columns of the first arithmetic structures while generating the first and second partial results.

8. (Currently amended) The method as recited in claim 5 wherein the generating of the first and second partial result is in response to execution of [[a]] the currently executing single arithmetic instruction.

9. (Currently amended) The method as recited in claim 6 wherein the generating of the first and second partial result is in response to execution of [[a]] <u>the currently executing</u> single arithmetic instruction.

10. (Previously presented) The method as recited in claim 1, wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of carry save adder tree columns.

11. (Previously presented) The method as recited in claim 1, wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of Wallace tree columns.

12. (Previously presented) The method as recited in claim 1, wherein at least one of the first and second pluralities of arithmetic structures is usable to perform both integer and XOR multiplication.

13. (Previously presented) The method as recited in claim 12, further comprising a logical circuit in at least one of the first and second arithmetic circuits supplying a fixed value if in XOR multiplication mode or a variable value that varies according to inputs supplied to the logical circuit if in integer multiplication mode, to thereby ensure a result is determined in XOR multiplication unaffected by carry logic performing carries in integer multiplication mode.

14. (Original) The method as recited in claim 13 wherein the logical circuit operates as a majority circuit in integer multiplication mode and outputs a zero in the XOR multiplication mode.

15. (Original) The method as recited in claim 1 wherein the first partial result is in redundant number representation.

16. (Original) The method as recited in claim 15 further comprising supplying the first partial result to an adder circuit to generate a non redundant representation of the first partial result and a carry out value.

17. (Original) The method as recited in claim 16 further comprising feeding back the carry out value to the adder circuit.

18. (Previously presented) The method as recited in claim 16, further comprising feeding back the carry out value to the second arithmetic circuit.

19. (Currently amended) The method as recited in claim 1, further comprising feeding back high order bits of the currently executing single arithmetic instruction from the first arithmetic circuit to the second arithmetic circuit for use with execution of a subsequent single arithmetic instruction of the processor instruction set.

20. (Original) The method as recited claim 1 further comprising storing the high order bits into an extended carry register.

21. (Currently amended) A method implemented in a device supporting a public-key cryptography application, the method comprising:

a first arithmetic circuit comprising a first plurality of arithmetic structures feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set in the public-key cryptography application, generated by the first arithmetic circuit to a second arithmetic circuit comprising a second plurality of arithmetic structures;

supplying a third number to the second arithmetic circuit;

the second arithmetic circuit generating a first partial result of a currently executing single arithmetic instruction of the processor instruction set in

the public-key cryptography application, wherein the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result being a representation of the high order bits summed with low order bits of a result of a first number multiplied by a second number and with the third number, the summing being performed during multiplication of the first number and the second number, the summing and at least a portion of the multiplication being performed in the second arithmetic circuit;

storing the first partial result; and

using the first partial result in a subsequent computation in the public-key cryptography application.

22. (Previously presented) The method as recited in claim 21, further comprising feeding back the high order bits through a register to the second arithmetic circuit.

23. (Currently amended) The method as recited in claim 21, further comprising:

the first arithmetic circuit generating a second partial result of the currently executing single arithmetic instruction, the second partial result representing the high order bits of the multiplication result of the first number multiplied by the second number.

24. (Currently amended) The method as recited in claim 21, further comprising:

generating a second partial result of the currently executing single arithmetic instruction, the second partial result representing the high order bits of the multiplication result of the first number multiplied by the second number summed with the high order bits of the previously executed single arithmetic instruction and the third number.

25. (Original) The method as recited in claim 24 further comprising supplying values generated in one or more most significant columns of the second arithmetic structures to one or more least significant columns of the first arithmetic structures while generating the first and second partial results.

26. (Currently amended) The method as recited in claim 23 wherein the generating of the first and second partial result is in response to execution of [[a]] <u>the</u> single arithmetic instruction.

27. (Original) The method as recited in claim 21

supplying the first partial result to an adder circuit to generate a non redundant representation of the first partial result and a carry out value.

28. (Original) The method as recited in claim 27 further comprising feeding back the carry out value to the adder circuit.

29. (Previously presented) The method as recited in claim 27, method further comprising feeding back the carry out value to the second arithmetic circuit.

30. (Previously presented) The method as recited in claim 21, wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of Wallace tree columns.

31. (Previously presented) The method as recited in claim 21, wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of carry save adder tree columns.

32. (Previously presented) The method as recited in claim 21, wherein at least one of the first and second pluralities of arithmetic structures is usable to perform both integer and XOR multiplication.

33. (Previously presented) The method as recited in claim 32, further comprising a logic circuit in at least one of the first and second pluralities of arithmetic structures supplying a fixed value if in XOR multiplication mode or a variable value that varies according to inputs supplied to the logical circuit if in integer multiplication mode, to thereby ensure a result is determined in XOR multiplication unaffected by carry logic performing carries in integer multiplication mode.

34. (Original) The method as recited in claim 33 wherein the logic circuit operates as a majority circuit in integer multiplication mode and outputs a zero in the XOR multiplication mode.

35. (Original) The method as recited in claim 21 wherein the high order bits are in redundant number representation.

36. (Currently amended) The method as recited in claim 21 further comprising feeding back high order bits of the currently executing <u>single</u> arithmetic instruction from the first arithmetic circuit to the second arithmetic circuit for use with execution of a subsequent single arithmetic instruction <u>of the processor instruction set</u>.

37. (Original) The method as recited in claim 21 further comprising storing the high order bits into an extended carry register.

38. (Currently amended) A processor configured to support public-key cryptography applications, comprising:

a first plurality of arithmetic structures configured to generate high order bits for an arithmetic ~~operation~~ <u>instruction of the processor's instruction set</u> in a public-key cryptography application that includes a multiplication operation; and

a second plurality of arithmetic structures configured to generate low order bits of the arithmetic ~~operation~~ instruction;

wherein the second arithmetic structures are further configured to receive the high order bits generated by the first plurality of arithmetic structures during execution of a previous instance of the arithmetic ~~operation~~ instruction in the public-key cryptography application and to generate a first partial result of a currently executing instance of the arithmetic ~~operation~~ instruction, wherein the arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result representing the high order bits summed with low order bits of a multiplication result of the multiplication operation; and

wherein the processor further comprises a register configured to store the first partial result for use in a subsequent arithmetic operation in the public-key cryptography application.

39. (Currently amended) The processor as recited in claim 38, wherein the first arithmetic structures are configured to generate a second partial result of the currently executing instance of the arithmetic instruction, the second partial result representing the high order bits of the currently executing instance of the arithmetic ~~operation~~ instruction.

40. (Previously presented) The processor as recited in claim 39, wherein the second arithmetic structures are further configured to supply values generated in one or more most significant columns of the second arithmetic structures to one or more least significant columns of the first arithmetic structures while generating the first and second partial results.

41. (Currently amended) The processor as recited in claim 39, wherein the first and second arithmetic structures are configured to generate the first and second partial results in response to execution of [[a]] an instance of the ~~single~~ arithmetic instruction.

42. (Previously presented) The processor as recited in claim 38, further comprising a register coupled to the first and second arithmetic structures to supply the high order bits to the second arithmetic structures.

43. (Previously presented) The processor as recited in claim 38, wherein the first partial result is in redundant number representation.

44. (Previously presented) The processor as recited in claim 43, further comprising an adder circuit configured to receive the first partial result and to generate a non redundant representation of the first partial result and a carry out value.

45. (Previously presented) The processor as recited in claim 44, wherein adder circuit is configured to feed the carry out value back to itself as an input.

46. (Previously presented) The processor as recited in claim 44, wherein adder circuit is configured to feed the carry out value back to the second arithmetic structures.

47. (Previously presented) The processor as recited in claim 38, wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of Wallace tree columns.

48. (Previously presented) The processor as recited in claim 38, wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of carry save adder tree columns.

49. (Previously presented) The processor as recited in claim 38, wherein at least one of the first and second pluralities of arithmetic structures is configured to selectively perform one of integer and XOR multiplication according to a control signal.

50. (Previously presented) The processor as recited in claim 49, further comprising a plurality of logic circuits in the first and second pluralities of arithmetic structures, each logic circuit responsive to the control signal to supply a fixed output value in XOR multiplication mode and a variable output value in integer multiplication mode, the variable output value varying according to values of inputs supplied to the logic circuit, to thereby ensure a result is determined in XOR multiplication mode unaffected by carry logic generating carries in integer multiplication mode.

51. (Previously presented) The processor as recited in claim 50, wherein the logical circuit is configured to operate as a majority circuit in integer multiplication mode and to output a zero in XOR multiplication mode.

52. (Previously presented) The processor as recited in claim 38, wherein the processor is a general purpose processor.

53. (Currently amended) A processor configured to support public-key cryptography applications, comprising:

a first plurality of arithmetic structures configured to generate high order bits for an arithmetic ~~operation~~ <u>instruction of the processor's instruction set</u> in a public-key cryptography application that includes a multiplication operation of a first and a second number; and

a second plurality of arithmetic structures configured to generate low order bits of the arithmetic ~~operation~~ <u>instruction</u>;

wherein the second arithmetic structures are configured to:

receive the high order bits generated by the first plurality of arithmetic structures during <u>execution of</u> a previous <u>instance of the</u> arithmetic ~~operation~~ <u>instruction</u>;

receive a third number; and

generate a first partial result of a <u>currently executing instance of</u> the arithmetic ~~operation~~ <u>instruction, wherein the arithmetic instruction does not include an explicit source operand for specifying the high order bits</u>, the first partial result representing the high order bits summed with low order bits of a multiplication result of the multiplication operation and with the third number; and

wherein the processor further comprises a register configured to store the first partial result for use in a subsequent arithmetic operation in the public-key cryptography application.

54. (Currently amended) The processor as recited in claim 53, wherein the first arithmetic structures are further configured to generate a second partial result of the <u>currently executing instance of the</u> arithmetic instruction, the second partial result representing the high order bits of the <u>currently executing instance of the</u> arithmetic ~~operation~~ <u>instruction</u>.

55. (Previously presented) The processor as recited in claim 54, wherein the second arithmetic structures are further configured to generate values in one or more most significant columns and to supply them to one or more least significant columns of the first arithmetic structures while generating the first and second partial results.

56. (Currently amended) The processor as recited in claim 54, wherein the first arithmetic structures are configured to generate the first and second partial result in response to execution of [[a]] <u>an instance of the</u> ~~single~~ arithmetic instruction.

57. (Previously presented) The processor as recited in claim 53, further comprising a register coupled to the first and second arithmetic structures to supply the high order bits to the second arithmetic structures.

58. (Previously presented) The processor as recited in claim 53, further comprising an adder circuit configured to receive the first partial result and to generate a non redundant representation of the first partial result and a carry out value.

59. (Previously presented) The processor as recited in claim 58 wherein the adder circuit is further configured to feed the carry out value back to itself as an input.

60. (Previously presented) The processor as recited in claim 58, wherein the adder circuit is further configured to feed the carry out value back to the second arithmetic structures.

61. (Previously presented) The processor as recited in claim 53, wherein at least one of the first and second arithmetic structures comprises Wallace tree columns.

62. (Previously presented) The processor as recited in claim 53, wherein at least one of the first and second arithmetic structures comprises carry save adder tree columns.

63. (Previously presented) The processor as recited in claim 53, wherein the arithmetic structures are configured to selectively perform one of integer and XOR multiplication according to a control signal.

64. (Previously presented) The processor as recited in claim 63, further comprising a plurality of logic circuits in at least one of the first and second pluralities of arithmetic structures, each logic circuit responsive to the control signal to supply a fixed output value in XOR multiplication mode and a variable output value in integer multiplication mode, the variable output value varying according to values of inputs supplied to the logic circuit, to thereby ensure a result is determined in XOR

multiplication mode unaffected by carry logic generating carries in integer multiplication mode.

65. (Previously presented) The processor as recited in claim 64, wherein the logical circuit is configured to operate as a majority circuit in integer multiplication mode and to output a zero in the XOR multiplication mode.

66. (Currently amended) An apparatus configured to support a public-key cryptography application, comprising:

means for feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set, generated by a first arithmetic circuit, to a second arithmetic circuit generating low order bits of a currently executing single arithmetic instruction of the processor instruction set;

means for using the second arithmetic circuit to generate a first partial result of the currently executing single arithmetic instruction, wherein the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result representing the high order bits of the previously executed single arithmetic instruction that are summed with low order bits of a multiplication result of a first number multiplied by a second number; and

means for using the first partial result in a subsequent computation in the public-key cryptography application.

67. (Currently amended) An apparatus configured to support a public-key cryptography application comprising:

means for feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set, from a first arithmetic circuit that generated the high order bits, to a second arithmetic circuit generating low order bits of a currently executing single arithmetic instruction of the processor instruction set;

means for supplying a third number to the second arithmetic circuit;

means for using the second arithmetic circuit to generate a first partial result of the currently executing single arithmetic instruction, wherein the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result being a representation of the high order bits of the previously executed single arithmetic instruction summed with low order bits of a result of a first number multiplied by a second number and with the third number; and

means for using the first partial result in a subsequent computation in the public-key cryptography application.